

Intel 8086 Microprocessor Memory Partition and Registers

Department of Computer Science & Engineering
BRAC University.

Course ID: CSE341
Course Title: Microprocessors

Lecture References:

? **Book:**

- ? *Microprocessors and Interfacing: Programming and Hardware, Chapter # 2, **Author:** Douglas V. Hall*
- ? *The 8086/8088 Family: Design, Programming, And Interfacing, Chapter # 2, **Author:** John Uffenbeck.*

Microprocessor

- **Intel 8086**

- The microprocessor 8086 can be considered to be the basic processor for the Intel X86 family from 1978.
- It has a 20-bit address bus along with 16-bit data bus.
- With the knowledge of 8086 16-bit processor, one can study the further versions of this processor 80286, 80486 and Pentium.

Memory Partitioning for 8086 Processor

? **The 8086 processor assign a 20-bit physical address to its memory locations.**

$2^{20} \rightarrow$ **1 Mbyte \rightarrow 1,048,576 bytes**

20 bits \rightarrow 5 hex digits

First addresses: 00000, 00001,...,0000A,...FFFFF.

But Registers are 16-bits (4 Hex digits) and can address only $2^{16} =$ 64 KBytes. 0000,0001,0002,.....FFFF

Suppose 20 bits \rightarrow 5 hex digits: 38A41H

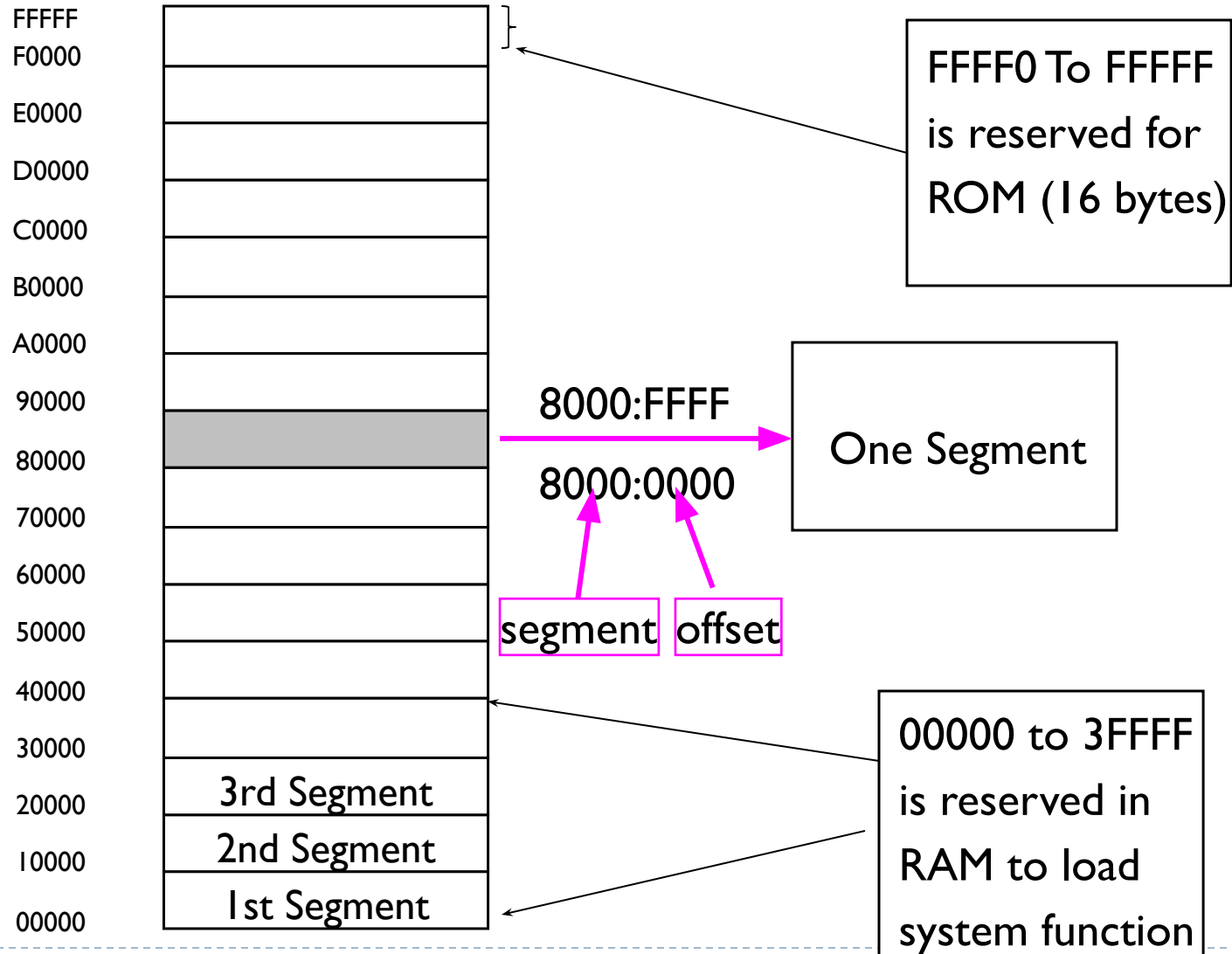
| | | | | |
|------|------|------|------|------|
| 3 | 8 | A | 4 | 1 |
| 0011 | 1000 | 1010 | 0100 | 0001 |

Here, 0011 1000 1010 0100 0001 = 20 bits = 5 Hex digits

? **Partition the memory into segments**

? Total of (16), 64 Kbytes segments might be positioned within the 1 Mbyte address space of an 8086 processor.

Memory Segment: *Address Space*



Segment Addressing

1st Segment →

- First Address - 00000h
- Last Address - 00000h + FFFFh = 0FFFFh

3rd Segment →

- First Address - 20000h
- Last Address - 20000h + FFFFh = 2FFFFh

Q) What is the second, third, second last, third last address for each of the above segments

Segment Addressing Continued

- 80001h → Segment Number, address number?
- BFFFDh → Segment Number, address number?

By address number, here it is meant first, second, third, last, second last, etc.

Memory Segmentation

- ? **Segmentation** is the process in which the main memory of the computer is **logically divided** into different segments and **each segment has its own base address**.
- ? It is basically used to enhance the speed of execution of the computer system, so that the processor is able to fetch and execute the data from the memory easily and fast.
- ? A segment is a logical unit of memory that may be up to 64 kilobytes long. Each segment is made up of contiguous memory locations.

Memory Segment: ***Address Space***

- ? ***Memory segment*** is a block of 2^{16} (64) KBytes consecutive memory bytes.
- ? Each segment is identified by a 16-bit number called **segment number**, starting with 0000 up to FFFFh. Segment registers hold segment number.
- ? Within a segment, a memory location is specified by giving an **offset** (16-bit) = It is the number of bytes from the beginning of the segment (0 → FFFFh).

Logical address is generated by CPU in perspective of a program.

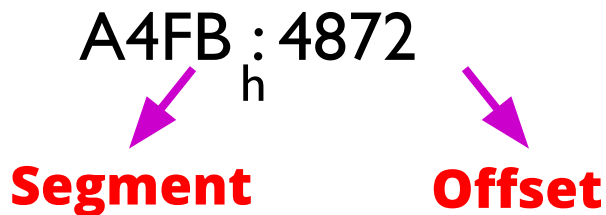
Physical address is a location that exists in the memory unit.

Memory Segment: ***Address Space***

- ? A memory location may be specified by a ***segment number*** and ***offset*** (logical address).

Example :

A4FB : 4872_h



Segment **Offset**

- ? **Offset:** is the distance from the beginning to a particular location in the **segment**.
- ? **Segment Number:** defines the starting of the segment within the memory space.

Segmented Memory Address



4Hex digits = $4 \times 4 = 16$ digits

so lowest value can be 0000 and highest value can be $FFFF$

Calculating Physical Address

Logical Address = (Segment number : Offset)

First / Starting Address of the segment = Segment number x 10h

The last digit(LSB) of first address of the segment has to be 0.

Physical Address = Starting Address of the segment + Offset

Example

Logical Address = (A012 : 0101)

Starting Address of the segment = $A012 \times (10)h = (A0120)h$

Physical Address = $(A0120)h + (0101)h = (A0221)h$

Example Math

Logical Address: (A4FB : 4872)

Segment No.: A4FB

Offset: 4872

First Physical Address: (A4FB0) h

Target Physical Address: (A4FB0) h + (4872) h = (A9822)h

Last Physical Address: (A4FB0) h + (FFFF) h = (BF4AF) h

2nd Last Physical Address: (A4FB0) h + (FFFF) h - 1 =
(BF4AE) h

3rd Last Physical Address: (A4FB0) h + (FFFF) h - 2
= (BF4AD) h

Example Math (Continued)

Second Physical Address: $(A4FB0)h + 1 = (A4FB1)h$

Third Physical Address: $(A4FB0)h + 2 = (A4FB2)h$

Starting Address of the next segment = Last address of this segment + 1 $= (BF4AF)h + 1 = (B4FB0)h$

Last Address of the next segment = Starting of the next segment + (FFFF)h $= (B4FB0)h + (FFFF)h = (C4FAF)h$

Hex digit Comparison

- Segment number/address - 4 hex digit
- First Address of segment - 5 hex digit
- Offset - 4 hex digit
- Physical Address - 5 hex digit

^All of the values has to be positive

Practice Question

Q1) Suppose, segment no = 1111 H, offset = 1332 H, calculate the physical address?

Q2) Suppose, physical address = 33330h, offset = 0020h, calculate the segment no?

Q3) Segment no. (CCCC)h. What is the base address, first physical address, second physical address, last physical address, second last physical address.

Practice Question

Q4) If segment number is (CCCC)h, if (CCCC2)h memory location has to be addressed, what should be the offset?

Q5) If segment number is (CCCC)h, if (BCCC2)h memory location has to be addressed, what should be the offset?

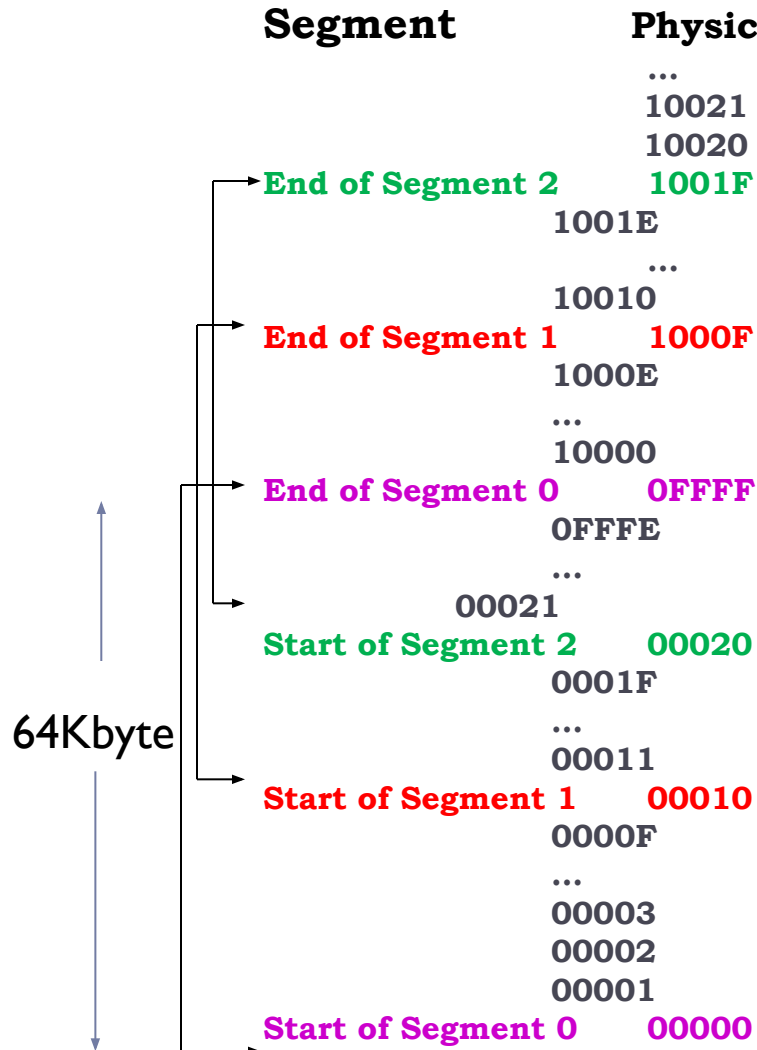
Memory Segmentation

? **Types Of Segmentation –**

- ? **Non-Overlapped Segment** – A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts before or after this 64kilobytes location of the first segment, then the two segments are said to be *Non-Overlapped Segment*.
- ? **Overlapping Segment** – A segment starts at a particular address and its maximum size can go up to 64kilobytes. But if another segment starts along with this 64kilobytes location of the first segment, then the two are said to be *Overlapping Segment*.

?

Physical Location of Segments



0001F physical address has
Different logical address

For Segment 0,
Base address = 0000
offset = 001F

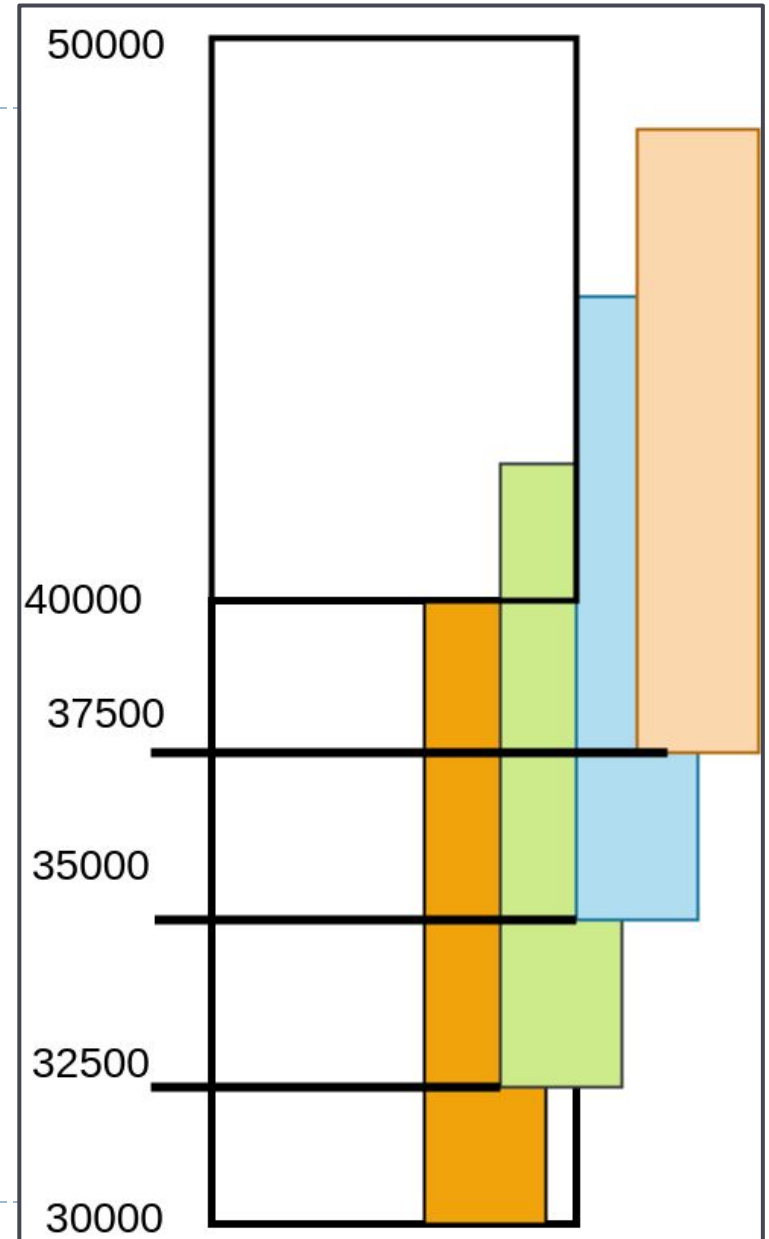
For Segment 1,
Base address = 0001
Offset = 000F

Overlapping Segmentation

How many different logical Address can you think of for one physical address? For eg: (38000)h

| <i>First Address of Segment</i> | <i>Offset</i> | <i>Segment No.</i> | <i>Offset</i> |
|----------------------------------------|----------------------|---------------------------|----------------------|
| 30000 | 8000 | 3000 | 8000 |
| 32500 | 5500 | 3250 | 5500 |
| 35000 | 3000 | 3500 | 3000 |
| 37500 | 0500 | 3750 | 0500 |

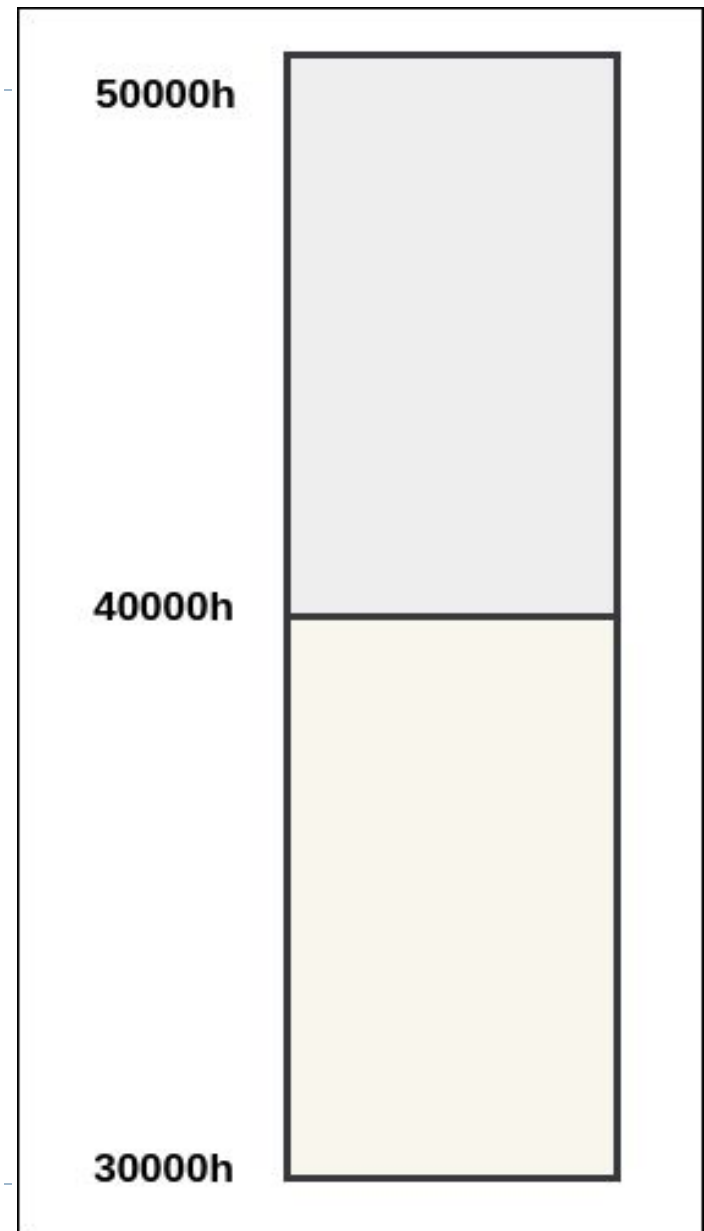
Each Colored box is a different 64kb segment.



Non - Overlapping Segments

How many different logical Address (segment number : offset) can you think of for one physical address? For eg: (38000)h

| <i>First Address of Segment</i> | <i>Offset</i> | <i>Segment No.</i> | <i>Offset</i> |
|----------------------------------------|----------------------|---------------------------|----------------------|
| 30000 0 | 8000 | 3000 | 8000 |



Question

- Can Non-overlapping segmentation have multiple logical address for the same physical address?
-

Memory Segmentation

? **Advantages of the Segmentation** The main advantages of segmentation are as follows:

- ? It provides a powerful memory management mechanism.
- ? Data related or stack related operations can be performed in different segments.
- ? Code related operation can be done in separate code segments.
- ? It allows to processes to easily share data.
- ? It allows to extend the address ability of the processor, i.e. segmentation allows the use of 16 bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20 bit registers.
- ? It is possible to enhance the memory size of code, data or stack segments beyond 64 KB by allotting more than one segment for each area.

Registers:

? **Registers:**

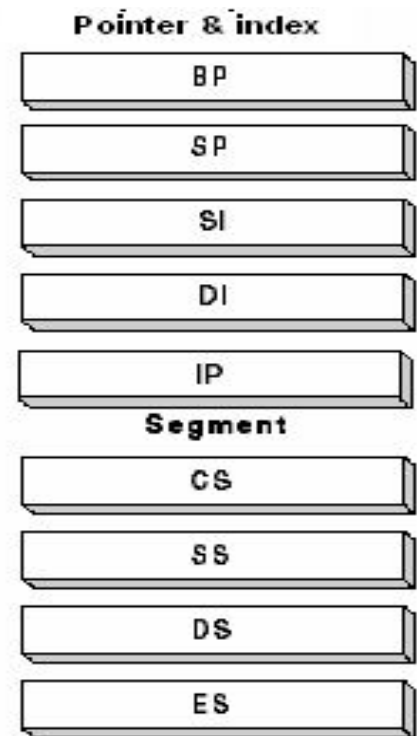
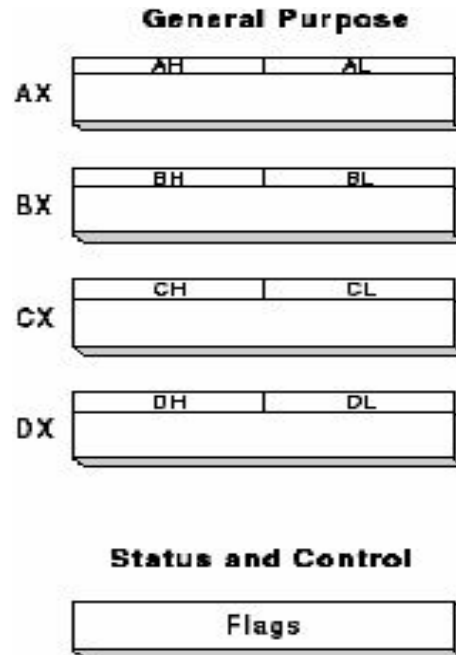
- ? **Information is stored in registers**
- ? **Registers are classified according to the functions they perform**
- ? **All Processors have internal registers some are visible and some are not visible to the programmers.**
- ? **Internal Registers are used as temporary storage for operands, and if the operand is already in memory, it takes less time for execution of the associated instruction.**

8086 Register Categories

? **Data registers (4):**
General data registers hold data for an operation (AX, BX, CX, DX).

? **Address registers (9):** (Segment, Pointer and Index registers) hold the address of an instruction or data.

? **Status register (1):**
FLAG register keeps the current states of the processor.



8086 Registers and Memory

Number of Registers: 14, each of that 16-bit registers

Memory Size: 1M Bytes

Registers of the 8086/80286 by Category

| Category | Bits | Register Names |
|-------------|------|------------------------------------------------------------------------------------|
| General | 16 | AX, BX, CX, DX |
| | 8 | AH, AL, BH, BL, CH, CL, DH, DL |
| Pointer | 16 | SP (Stack Pointer), Base Pointer (BP) |
| Index | 16 | SI (Source Index), DI (Destination Index) |
| Segment | 16 | CS (Code Segment) DS (Data Segment) SS (Stack Segment) ES (Extra Segment) |
| Instruction | 16 | IP (Instruction Pointer) |
| Flag | 16 | FR (Flag Register) |

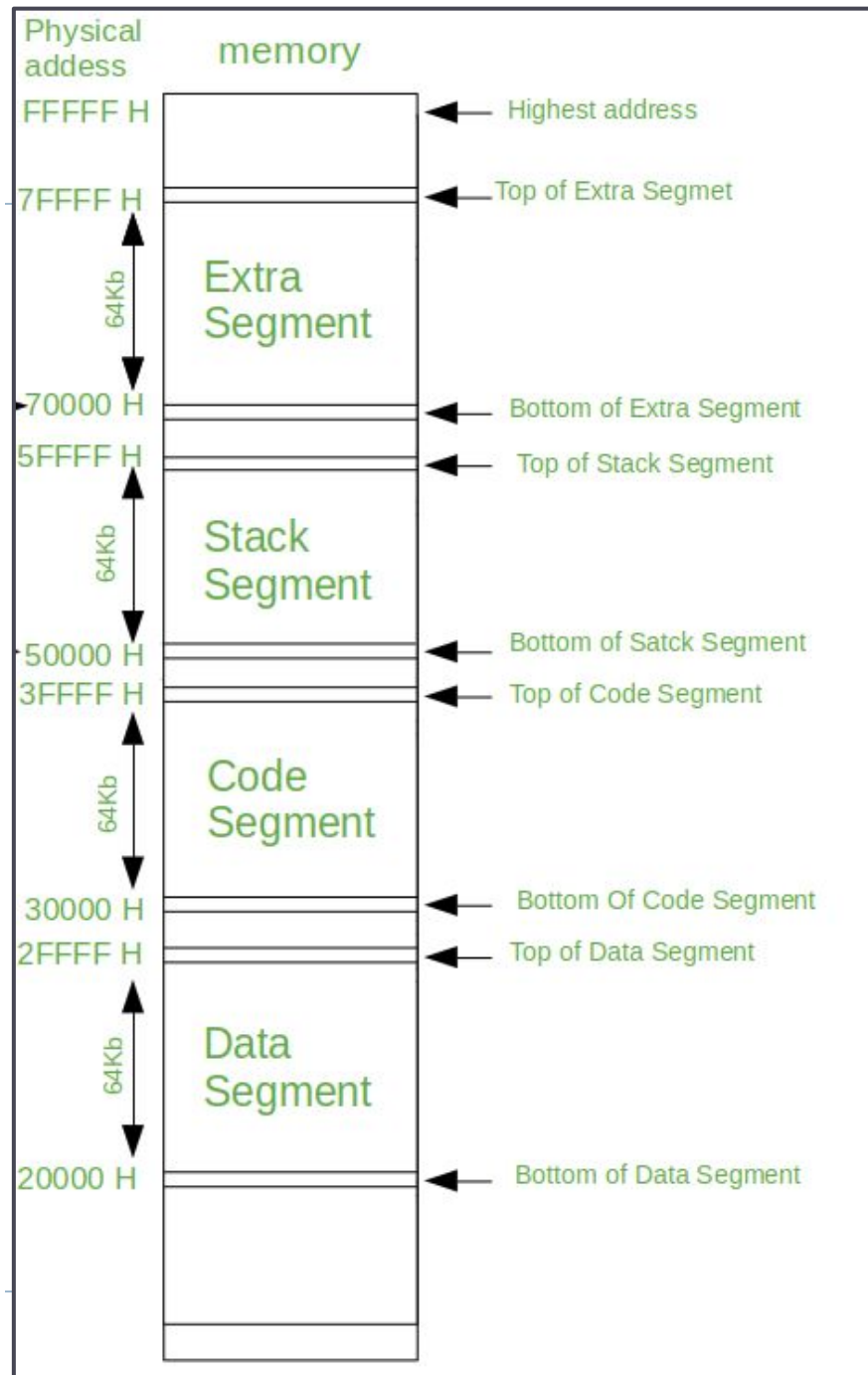
Memory Segment and Segment Registers

Note that, 8086 does not work with the whole 1MB memory at a given time, it works only with four 64KB segments within the whole memory

The following registers holds the ***segment number*** of specific segments in the memory:

- **Code segment register (CS):** holds segment number of the code segment.
- **Data Segment register (DS):** holds segment number of the data segment.
- **Extra Segment register (ES):** extra segment : holds alternate segment number of the data segment.
- **Stack Segment register (SS):** holds segment number of the stack segment and used when sub-program executes.

This is
Inside
Memory
(RAM)



Finding Logical Address of an Instruction

Instruction at memory location (30500) h

Code segment has a base address of (30000)h

So offset - 500

Register Value:

CS: 3000

IP: 500

General Data Registers

- ? These are 16-bit registers and can also be used as two 8 bit registers: ***low and high bytes*** can be accessed separately
- ? **AX (Accumulator)**
 - ? Can be partitioned into AH,AL
 - ? Most efficient register for arithmetic, logic operations and data transfer: the use of AX generates the shortest machine code.
 - ? In multiplication and division operations, one of the numbers involved must be in AL or AX
- ? **BX (Base)**
 - ? The base address register (offset)
- ? **CX (Counter)**
 - ? Counter for looping operations: loop counter, in REP instruction, and in the shift and rotate bits
- ? **DX (Data)**
 - ? Used in multiply and divide, also used in I/O operations

Pointer and Index Registers

- ? **Used for offset of data, often used as pointers. Unlike segment registers, they can be used in arithmetic and other operations.**
- ? **SP (Stack Pointer):**
 - ? Used with SS for accessing the stack segment.
 - ? Holds **Offset** address relative to SS
 - ? Always points to word (byte at even address)
 - ? An empty stack will had SP = FFFEH
- ? **BP (Base Pointer):**
 - ? Used with SS to access data on the stack. However, unlike SP, BP can be used to access data in other segments.
 - ? Primarily used to access parameters passed via the stack
 - ? Holds **Offset** address relative to SS

Pointer and Index Registers

? **SI (Source Index):**

- ? Source of string operations. Used with DS (or ES).
- ? Can be used for pointer addressing of data with effective address (EA)
- ? Used as source in some string processing instructions
- ? Offset address relative to DS

? **DI (Destination Index):**

- ? Destination of string operation. Used with ES (or DS).
- ? Can be used for pointer addressing of data
- ? Used as destination in some string processing instructions
- ? Offset address relative to ES

? **IP (Instruction pointer):**

- ? Points to the next instruction.
- ? Offset address relative to CS

Memory Segmentation

? **Rules of the Segmentation:** Segmentation process follows some rules:

- ? The starting address of a segment should be such that it can be evenly divided by 16.
- ? Minimum size of a segment can be 16 bytes and the maximum can be 64 kB.

| Segment | Offset Registers | Function |
|---------|------------------|----------------------------------------------------------|
| CS | IP | Address of the next instruction |
| DS | BX, DI, SI | Address of data |
| SS | SP, BP | Addresses in the stack |
| ES | BX, DI, SI | Address of destination data (for string instructions) |

IF both DS and ES present, take DS for offset BX, DI, SI

For Practice

| | | | | | | | | |
|---------|-------|-------|-------|-------|-------|-------|-------|-------|
| Address | 10600 | 10601 | 20100 | 20101 | 20600 | 20601 | 30600 | 30601 |
| Data | 12h | 34h | 56h | 78h | 10h | 20h | 55h | 32h |

Given DS = 2000h, SS = 1000h, CS = 3000h, BP = 0500h, SI = 0100h.

Q1) MOV BL, [DS + SI]

Q2) MOV BH, [SI]

Q3) MOV BX, [BP + SI]

Q4) MOV BX, [BP+SI] 100h

Worked Example

| | | | | | | | | |
|---------|--------|--------|--------|--------|--------|--------|--------|--------|
| Address | 11000h | 11001h | 20400h | 20401h | 25500h | 25501h | 20600h | 20601h |
| Data | 01h | 23h | 63h | 45h | 98h | 76h | 34h | 73h |

L

H

DS = 2000h SS = 1000h DI = 0800h

MOV AX, [SI]. AX = 7334h

Find SI?

$$DS * 10 + SI = 20600h$$

$$2000 * 10 + SI = 20600h$$

$$SI = 0600h$$

Worked Example

| | | | | | | | | |
|---------|--------|--------|--------|--------|--------|--------|--------|--------|
| Address | 11000h | 11001h | 20400h | 20401h | 25500h | 25501h | 20600h | 20601h |
| Data | 01h | 23h | 63h | 45h | 98h | 76h | 34h | 73h |

L **H**

DS = 2000h SS = 1000h DI = 0800h

MOV BL, [BP + DI]. BL = 01h

Find BP?

$$SS * 10 + BP + DI = 11000h$$

$$1000 * 10 + BP + 0800h = 11000h$$

$$BP = 0800h$$

Thank You !!

